Hierarchical Organization of Certification Authorities for Secure Environments

Lourdes López, Justo Carracedo
Departamento de Ingeniería y Arquitecturas Telemáticas (DIATEL)
EUITT Universidad Politécnica de Madrid
Ctra. de Valencia, Km. 7 28031 MADRID ESPAÑA
TF: 34-1-336 78 18 FAX: 34-1-336 78 17
e-mail: lourdes.lopez@diatel.upm.es
e-mail: carracedo@diatel.upm.es

Abstract

This paper presents a model of hierarchical organization of Certification Authorities which can be applied to any open system network. In order to study the feasibility of the proposed model, a pilot experiment within a university environment is being carried out. The authors of this paper have developed an application which provides the users with security services using X.509 certificates. The authors have also developed a security server to provide RSA keys and management of certificates.

The hierarchical infrastructure that is being created needs a multi-level policy which implies the use of various types of documents which are managed by people with different roles. One of the objectives being followed is to formalize the treatment of the information about policy, for which some components of the extensions field of the version 3 certificate have been used and other new ones are proposed.

1. Introduction

The current expansion of computer networks and the opening that their interconnection produces, especially through the Internet, means that a large part of the information that their users handle is continuously traveling through the network and crossing numerous intermediate systems. This means that the number of possible attacks and illegal operations which can be produced increases dramatically. In order to protect the networks from these dangers it is necessary to fit them with a series of security services [6] which guarantee the identity of the communicators, the integrity of the information which is transferred, and in some cases their confidentiality as well. These services of authentication, integrity and confidentiality can be provided using cryptosystems based on public key methods. In these cryptographic systems, the encryption and decryption operations are carried out using a pair of keys: the public key and the secret key.

In public key cryptosystems [12], the secret key must remain hidden in the owner's domain; in this way the danger of interception by an attacker is reduced. Nonetheless, the public key can be widely known. However the need for entities to make their public key known poses new problems.

In the first place, mechanisms to publish these keys must be found. Initially, the Directory [7] was thought of as an ideal place for its publication, but its low deployment means that the use of e-mail and web servers which can also perform this task are beginning to be considered.

Second, a scheme which guarantees the ownership and the validity of the public key must be followed. In the most basic scheme the entities directly have confidence in the validity of the public key, which is provided in a one to one exchange by the other communication entity. The assurance scheme is improved when it is based on the trust in a third party. In this case each entity must only confide in a Trusted Third Party (TTP). When this scheme is applied to a security infrastructure based on public key techniques, the TTP is denoted a Certification Authority (CA). This authority certifies that this key is valid for a determined period. The CA emits and signs a digital document, valid for a period of time, which

associates the identity of an entity with its public key. This document, called a certificate, is itself secure as it is digitally signed by the CA. Certificates are made public.

When a community of users grows, a single CA can become overloaded by the number of certificates it has to manage. Moreover, in most cases, each company or organization wants to have control over the way in which its users generate the keys, the validity period of the certificates, their possible fraudulent use, etc. This makes the distribution of the certification functions convenient, giving place to the appearance of various CAs, whose security policies might be different.

In an assurance scheme based on TTPs, when an entity A needs to obtain the public key of another entity B, either to send an encrypted message to B or to validate the signature of B, A can obtain the aforementioned key from a certificate, thus checking its validity. To do that, A needs to check that the certificate is really signed by the CA that issued it, for which it is necessary to obtain the public key of the CA.

When multiple CAs exist in a security domain, if the entities that trust different CAs want to communicate amongst themselves, the CAs must organize themselves to certify each other in such a way that, between them, there is trust. In this case, for the entity A to reliably know the entity B's public key, it has to have an arranged set of certificates made up of entity B's certificate and all the certificates of the various CAs involved, until reaching a CA in which A trusts. This set of certificates, in order, is known as a *certification path.*

2. Platform used in the experiment

The university environment where the experiment is being developed is a Technical School that currently has 2,000 students. Professors, students and administrative staff use various telematic services and applications in order to communicate with each other (e-mail, ftp, ftam, www, etc). This School has various interconnected local area networks and uses the Internet to communicate with the Central Staff of the university and with the rest of the world. Some information transferred in this environment needs to be protected by applying authentication and integrity services and, in some cases, confidentiality as well.

In order to provide these security services, a user's tool called SecKit and a security server named SecServer have been developed by the authors, within the framework of a European project.

2.1. SecKit: a tool to provide the users with security services

SecKit allows different security mechanisms to be applied to end-to-end applications (for example, e-mail, ftp and ftam), following the X.509 recommendation. It also allows direct and automatic access to the SecServer.

The operations which can be carried out from SecKit are the following:

- Generation of keys

  SecKit allows the generation of DES and RSA keys.

- Sending of secure files

  SecKit allows three different levels of security on a file:

  - Signed File: the file is digitally signed using a hash function (md2, md4 or md5) and the secret (RSA) key of the signer.
  - Encrypted File: the file is encrypted using a symmetrical key through the DES algorithm, or the public (RSA) key of the receiver (just for critical and small information).
  - Secure File: a symmetric key which is used to encrypt the file using DES is automatically generated. The generated symmetric key is encrypted with the public (RSA) key of the receiver and is added to the file. The digital signature of the original file and the X.509 distinguished name (DN) of the signer is added. The three kinds of secure files are coded in ASN.1 so that they can be interpreted openly by other applications which have a parser for ASN.1. The definition of Secure File is original, while that of Signed File and Encrypted File are taken from the X.509 recommendation. If the file has to be transferred in ASCII format, SecKit allows provides the necessary encoding.

- Reception of secure files

  SecKit allows files to be decrypted and signatures to be validated, thus obtaining the original file or informing about detected problems.

- Access to the Directory

  SecKit allows a user who has an entry in the Directory to store his/her certificates in it, and also allows a user to extract a certificate from the entry of another user.

The first version of SecKit, which appeared towards the end of 1993, was developed for Sun computers with the operating system UNIX. Its user interface was not very user friendly and the cryptographic module was implemented in software, so the problem of secure storing of secret keys arose. The certificate used was the X.509 certificate version 1 [7].

As a result of work carried out in another project, a new version for PCs is also now available. The user interface has been improved, both in the version for Sun computers and for PCs, offering a very similar interface in X-Windows and in MS-Windows respectively (see Figure 1). The cryptographic module has been replaced by a hardware board connected to a smartcard, thus solving the problems of storing the secret key and controlling access.

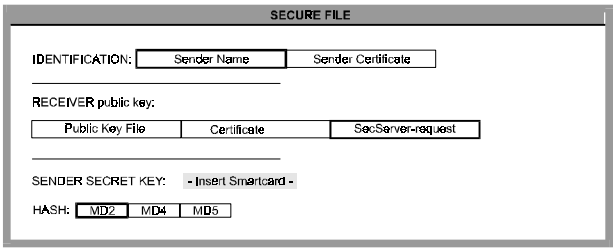

Figure 1: SecKit user's interface

SecKit is not only a user's application; its application interface (API) allows different functions of SecKit to be accessed by other applications. Following this line of development, SecKit was used to implement a secure mail application compatible with PEM.

Currently, work is being carried out on SecKit to substitute version 3 certificates for version 1 certificates [8] [4] [9] and to apply the functionalities offered by the API of SecKit in order to make secure Web pages.

2.2. SecServer: a security server

From an external point of view, SecServer is a server of RSA keys and of X.509 certificates. Internally it also includes the functionality of a Registration Authority and of a Certification Authority.

The services that SecServer offers are the following:

- Registration of end users or of CAs which need to be certified by the CA associated with the server. If the users do not have entries in the Directory, SecServer assigns them DNs.
- Generation of RSA keys for users who ask for them. The secret key is sent to the applicant, encrypted with another key which is sent to the user through an off-line mechanism or which is recorded on the user's smartcard. The public key is sent certified.
- Generation of a public key certificate for a user and sending of the certificate to the applicant.
- Renewal of expired certificates.
- Sending of a requested certificate, either of the requesting user or of another user.
- Sending of the Certificate Revocation List (CRL) which the associated CA has created for the certificates issued by it; and processing a request for a certificate's revocation.

The communications protocol with SecServer is very simple and open. A request for a service is made by sending an e-mail message, indicating in the first line of the body the type of service requested. Each request uses a phrase which follows a friendly and easily recognizable syntax. If the requested service requires some type of additional information to be sent, this is added to the message's body after the request line. An example is given in Figure 2.

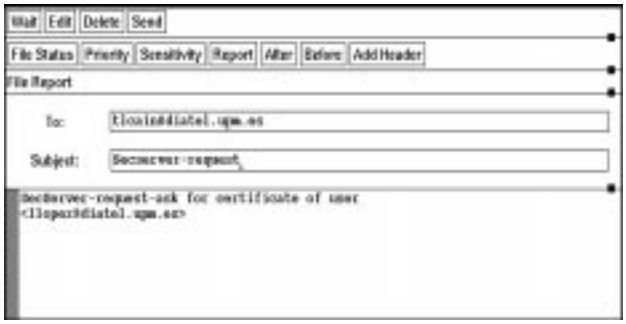

Figure 2: Request for service to SecServer

The SecServer responds by sending what has been requested, for example a certificate, a key or a CRL, by e-mail in ASCII format. If the user wants to receive the certificates, the keys and the CRLs directly in ASN.1 format, when the user is enrolled he/she can request that the SecServer send them using FTP or FTAM. The confirmation that the user has been accepted, the confirmation that the CA has revoked a certificate and the CRL checking are also sent by e-mail. In the same way, in the case of not having been able to process a

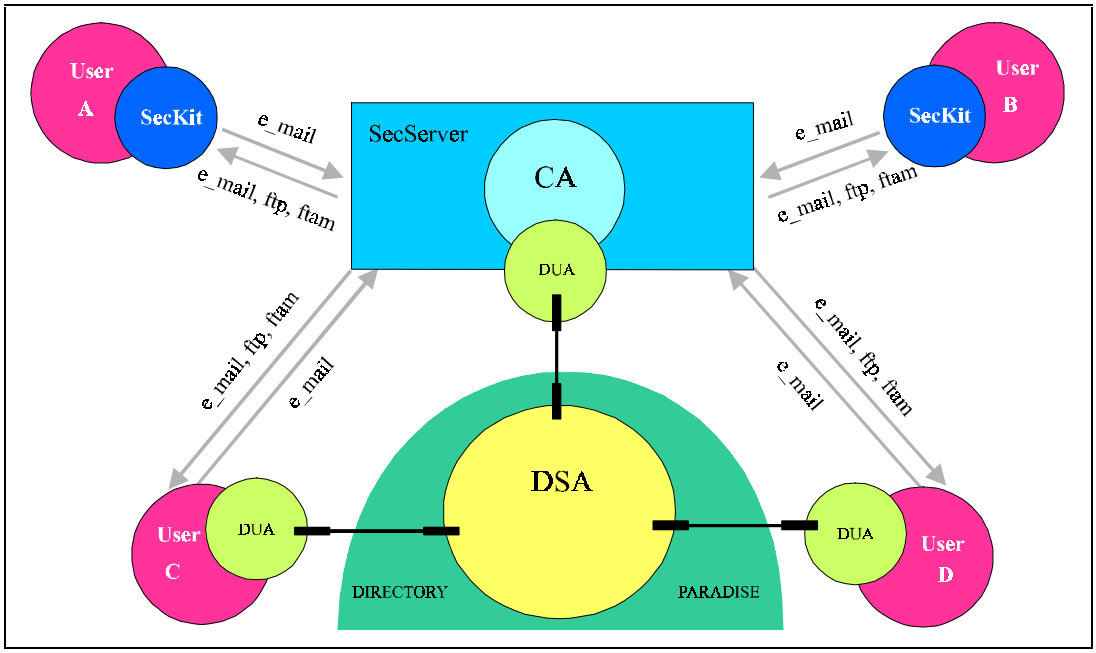

Figure 3: Experiment environment

request, SecServer also emits a message indicating the cause for failure.

Work is now being done on an on-line connection between SecKit and SecServer through a Client/Server architecture. This will allow SecServer response times to be much faster. This new type of on-line access will not replace access by e-mail, rather it present a compatible alternative.

SecServer has direct access to the X.500 PARADISE Directory (see Figure 3), thus it is able to obtain the certificate of any user who has stored it in the Directory, whether that certificate was issued by the CA associated with SecServer or by another CA. As well as access to the Directory, SecServer has a local data base in which the certificates issued by the CA associated with it are stored. This data base allows the location of certificates not only based on the user's DN, but also based on other names which can uniquely identify the user, for example his X.400 or rfc822 mail address, his DNS, his IP name, or his EDI identifier.

This allows the user to provide certificates of other users that do not have access to the Directory and that have been certified by their own CAs.

SecServer can be used automatically from SecKit (see Figure 3), but also manually using e-mail (see Figure 3), meaning that can act as an open server to any user who has either X.400 or SMTP electronic mail.

3. Certification hierarchy

The trust scheme used in this experiment is based on trust in CAs following the X.509 recommendation. This recommendation, which defines a secure model in order to provide the authentication service to X.500 Directory users, has been the basis for the majority of proposals for application security services in recent years.

The greatest contribution of X.509 is the format of the [7] certificate, the terminology introduced and the authentication environment presented. X.509 expresses the need to establish relationships between CAs but does not indicate the way this must be done.

Study of existing organizations, real life experience and the hierarchical character of the University in which the experiment has been carried out, have all led to the decision to adopt a hierarchical model.

The Internet community adopted a hierarchical organization model for its secure electronic mail PEM [5]. In this hierarchy there exists one top level CA (TLCA) named the Internet Policy Registration Authority (IPRA). The IPRA is in charge of defining a general policy for all of the Internet, and of registering and certifying a second level of CAs named Policy Certification Authorities (PCAs). The PCAs establish and publish special policies for the registration of users and organizations. PCAs certify other CAs which certify end users and other CAs which are organizationally

subordinate to them. Under PCAs three different types of CAs exist; these are: organizational CAs (OCAs) which offer their services to users and to other CAs subordinate to them; residential CAs which offer their services to users who are not affiliated to any organization; and PERSONA CAs which offer their services to users who want to remain anonymous.

In the model adopted by PEM, although use of the X.500 Directory is not required, a naming system is used. An isomorphism is imposed with the Directory hierarchy from the first level of CAs under the PCAs. In this way, a CA must be an organization or an organizational unit, and the Distinguished Name (DN) of a user and CA that is underneath must be a subset of the name of the CA that certifies it.

The model of hierarchical organization put forward by PEM offers numerous advantages to its users when it comes to validating certificates, but the establishment of such a strict hierarchy brings new problems. In the first place, name subordination causes a series of problems. For example, if a CA and its users want to operate under different policies, they need to have different keys for each policy. If a user or a CA, certified by one CA, wants to be certified by a different CA, it has to be identified with two different names. In the second place, the existence of only one TLCA in the world, resident in the U.S., creates distrust when it comes to being accepted as the only alternative in the European Union.

The European PASSWORD project [11] initially proposed a hierarchy for the European scientific and research community, in which each country could have its own TLCA. Each country's TLCA could certify different PCAs. In turn, the PCAs could certify organizational and PERSONA CAs, the concept of a residential CA not existing. From this level down, the model is the same as PEM model. Even though the PASSWORD design was different to the PEM design, the difficulties found when putting this model into practice lead to PASSWORD adopt a PEM style model with a single TLCA.

In order to define the model of CA organization presented in this paper, the models in addition to PEM and PASSWORD were studied and analyzed [3] this included the model that ANSI presents in the X9.57 document [1], the model of the National Institute of Standards and Technology (NIST) [10], and the model adopted by Pretty Good Privacy (PGP) [14]. The latter, although it does not represent a TTP model nor a hierarchical model, was considered important to analyze due to its popularity.

The proposed hierarchical organization model has various aspects which differentiate it from the analyzed models. In fact, the majority of these models can be considered as particular cases of this new, more general model.

The proposed model, which is illustrated in Figure 4, is as follows:

- The general model is composed of multiple trees which can cross-certify amongst themselves. The arrows indicate the sense in which the certification is made.
- The top of each tree is formed by a compound node named Policy CA Set (PCAS) which defines the policy of a security domain belonging to a company, an institution, or any "group" which wants to be managed by communal rules. This "group", which can have national, continental or universal scope, will be denote in this model a Communal Security Rules Group (CSRG). The CSRG can be formed by entities belonging to the same administrative organization (for example, a company or a university) or by entities belonging to different organizations which need to intercommunicate (for example, by Internet electronic mail).
- The PCAS node is made up of two different types of CAs. All of them define aspects of the CSRG policy, but the CAs that are lower in the hierarchy can only restrict the policy that is defined above them. The expansion of a policy by a hierarchically inferior CA would involve the creation of a new policy and therefore a new PCAS node. The two types of CAs in a PCAS node are:
  - The top level CA of the node and therefore the top of the CSRG tree is the Group CA (GCA). The GCA is unique for each tree and defines the general policy associated with that CSRG. This CA must always exist in a scheme with TTPs.
  - Optionally one or more levels of Subgroup CA (SGCA) can exist under the GCA. These CAs represent subgroups within a CSRG which have a certain autonomy of policy and therefore can restrict the general policy defined by the GCA. For example, national restrictions in the case of groups made up of members from different countries.

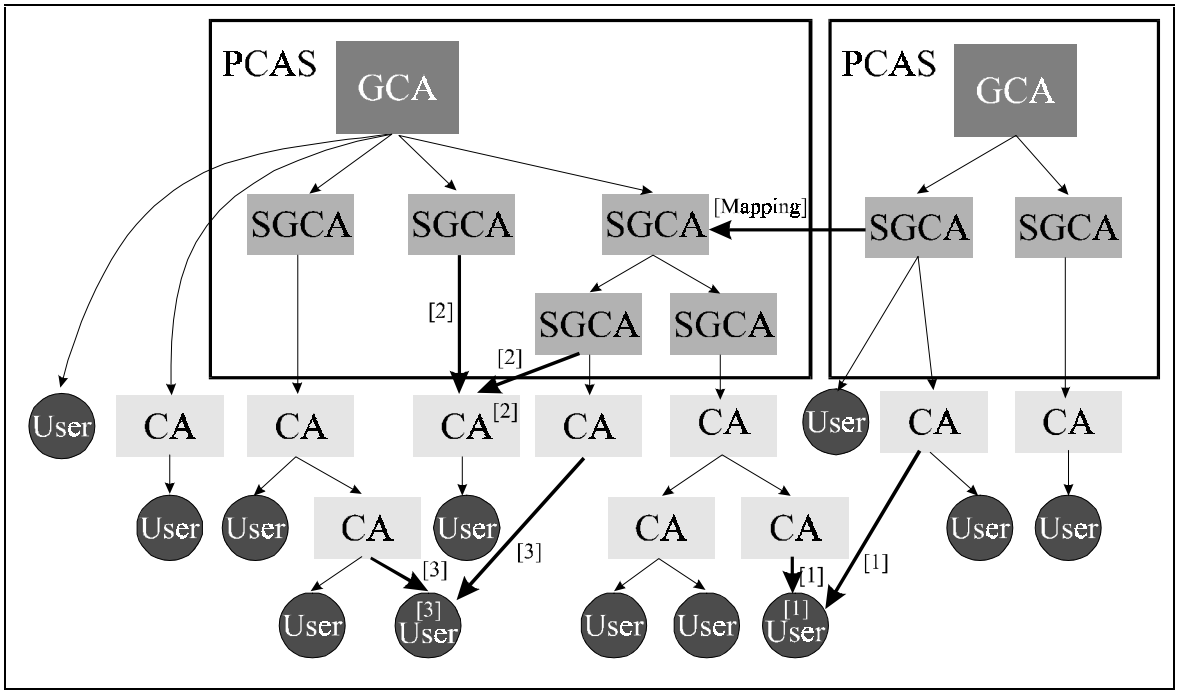

Figure 4: Model design

Under the PCAS node are normal CAs which do not define aspects of policy. The functions of these CAs are to register users and to generate certificates in a distributed way according to the needs of the CSRG. These CAs will assume the policy defined by the superior PCAS. Depending on this policy, the CAs below the PCAS will be able to be classified as CAs of different kinds. Different types of CAs can take into consideration special registration characteristics or roles of their users. This is the case recognized the SET model of Visa and MasterCard [13], in which three types of CAs are: the CA Cardholder (CCA) for Visa and MasterCard card holders, the Merchant CA (MCA) for traders who make sales with Visa and MasterCard, and the Payment CA (PCA) for intermediary banking groups.

Under any CA, whether it is a normal CA or a PCAS node CA, are end users.

3.2. Definition of security policies in the model

A *security policy* can be defined as the set of rules, obligations, rights and responsibilities that must be accepted and followed by the members of a security domain. When this set of rules, obligations, rights and responsibilities refers to the use of certificates and of certified public keys, the security policy is called a *certificate policy.*

A security policy, and in particular a certificate policy, is made up of a set of policy elements, each relating to an aspect of that policy. A complete list of the security elements that can make up a policy does not yet exist, but the different policies already defined reflect a series of elements in common.

After carrying out a study of different security policies [2] and analyzing the needs of the certificate policy for the experimental domain it was concluded that the following policy elements must be considered:

- *Procedures for the registration of users.* The mechanisms for the identification and acceptance of end users and CAs in a CSRG must be defined. Name subordination rules, if desired, can also be defined. For example, user registration may require a guarantee or identification with a document such as a passport, or may be able to be done through e-mail.
- *Protection mechanisms.* The software and hardware requirements needed to protect the CSRG infrastructure must be defined. For example, it can be requested that the certificate generation software be in a machine not connected to the network, that the secret key of the CAs be in a smartcard, and that the users' secret keys be on protected floppy disks.
- *Key management procedures.* Aspects of the

generation, distribution, and validity of keys must be defined. For example, it can be specified whether the keys can be generated by software or hardware mechanisms, whether they can be generated by the users or by their CAs, and in the latter case, the distribution mechanism, the size of the keys, and their validity period.

- *Certificate management and verification procedures.* Aspects related to the validity, renewal and revocation of certificates and to the verification of other users' certificates must be defined. For example, it can be established that the validity period be fixed for all the certificates of the domain, be fixed by each CA, or be requested by the user.
- *Audit and file.* The means used to audit and file the events and information necessary to guarantee the protection of the CSRG must be defined.

Another aspect that must be studied in order to define a policy is the *specification of the use of keys.* The possible uses of the keys by CAs or end users must be defined. Many kinds of documents, each with different security requirements, may be managed in a security domain. Also, entities with different roles in a CSRG may be authorized to manage different types of documents.

It may therefore be necessary to define the different roles, the different types of documents, the different uses that can be made of the keys, and the relation that exists between these three concepts. Specifically, for each role, the type of documents it can manage and the level of security that each document needs must be indicated. Also, the compatibility or incompatibility between different roles must be indicated. For example, in a CSRG in which the only type of documents are letters and in which roles between users are not differentiated, it is enough noting the uses of the keys. In the CSRG of an University, people have different roles such as teachers, students, members of management or administration, and other staff. Different types of documents also appear, for example, registrations, qualification certificates, and requests for equipment, each needing a different level of security. For example, a qualification certificate can only be handled by teachers and administration staff and requires a high level of security, such as, various signatures, confidentiality, integrity, authentication and non-repudiation of destination and origin. Furthermore, depending on the circumstances, a person may have more than one role and certain roles may be incompatible. For example, a person can never be a teacher and a pupil at the same time in this university school.

In the model presented in this document, the security policy is based on the use of public keys and X.509 certificates, therefore it is a certificate policy. One of the objectives pursued by the authors work group is to define a model in which the identification of the certification policies is carried out automatically. For this reason work is being done with the X.509 version 3 certificate and the standard extensions, which are now agreed standards texts [9].

In this model, each certificate policy is defined in a PCAS node and represents the policy of a CSRG. Each policy must be accepted and followed by all the nodes linked to a tree whose root node corresponds to a GCA. Thus, when a CA or a user is a registered member of a CSRG, it will have a certificate in which the identifier of the CSRG's policy appears in the *certificatePolicies* extension field. This field can be critical or not, depending on the issuing CA.

Thus, each PCAS node represents a single general policy but as was indicated when introducing the model, the PCAS node can contain SGCA nodes which represent subgroups with a certain refinement of the policy within the CSRG. Each SGCA can define a subpolicy of the general policy, but it will always be restricted form of the general policy. In order to define these subpolicies, the extension field *policyConstraints* is used. In summary, a PCAS node represents a single policy and various subpolicies which are restrictions of the general policy defined by the CA root of the node. The nodes that form a part of the "supernode" PCAS cannot be certified by the CAs of another tree, as this would mean that they accept another different policy from the one they are defining.

In this model, a node which represents an end user or a normal CA which does not belong to the PCAS node can be certified by more than one CA. This means that the node can act under as many policies as it has certificates. Thus, a node of this type can be certified by CAs which belong to a different tree, which means that the node belongs to more than one CSRG (arrows [1], Figure 4), or it can be certified by CAs of the same tree but which belong to a different subtree under the PCAS node, and so the node would belong to more than one subgroup within the same CSRG (arrows [2] [3], Figure 4). In this model, for a node to be able to act under different policies, it needs a multi-policy certificate or different certificates. The use of different certificates does not necessary implied the use of different public keys. The same public key certified under different policies would enable one user to belong to various groups. The user would only need to possess more than a couple of RSA

keys if any of the policies had any requirement which forced the user to generate another pair of keys.

Therefore in the proposed model, the tree concept is linked to the policy concept of a CSRG, which is determined by the PCAS node. The concept of belonging to a tree is linked to the certificate and not to the subject entity of the certificate. That is to say, one entity (normal CA or end user) can belong to more than one CSRG, or to more than one subgroup within the same CSRG. Depending on the certificate it uses, it is recognized as a member of one group or another.

One need which this model has is that, on occasion, various members of different CSRGs have to establish joint communications. However, there is no need to create a new CSRG if the policies of the groups are equivalent and the establishment of communication between the interested parts is allowed. For example, suppose teachers of various Universities, each of which has its own policy, are going to work on the same research project. This is when the mapping concept can be used. A CA from the domain to which the user does not belong issues a certificate for the CA that certifies the user in his/her CSRG, indicating in the field *policyMappings* that the issuing CA's policy is equivalent to the subject CA's policy. Since, in this model, the only CAs that define policy are those of the PCAS node, this mapping has to be established at this node's level ([Mapping], Figure 4).

A subject which is currently being studied and for which this model does not provide a solution is the location of the distribution points of CRLs. These points must be established in such a way that the revocation of certificates and the reference to the CRLs is as efficient as possible. Currently, in the experimental platform that is being used, each CA stores its CRLs but the reference and updating mechanisms are very slow.

3.3. Establishment of the level of assurance

In the presented model, an assurance scheme based on CAs is used, in which one entity (entity [1211], Figure 5) has confidence in the validity of the public key of other entity (entity [2112], Figure 5) if a CA of mutual trust exists (node [21], Figure 5). For the certificate of an entity to be considered valid by another entity, the latter needs to verify that the signature of that certificate is valid. In order to do this, it is necessary to obtain the

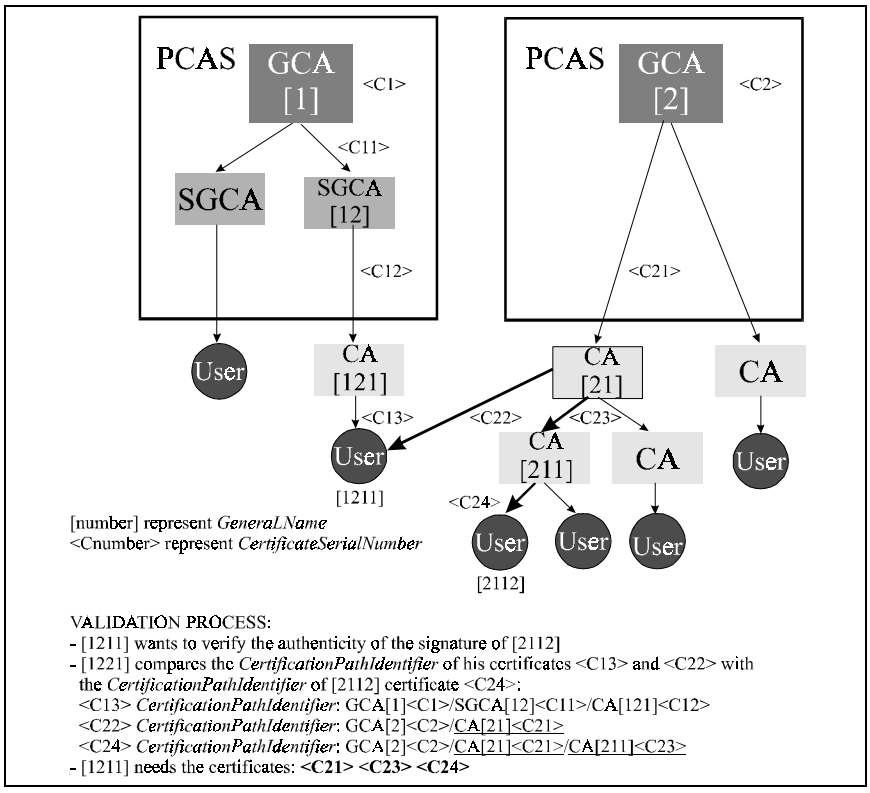

Figure 5: Example of certification path

certificate of the public key of the issuing CA. If the entity that is carrying out the validation process does not reliably know the public key of this CA, it will also need to check the validity of the certificate of this CA's public key, and so on until it reaches a CA in which it trusts, which will correspond with the common node in the organization model. The set of certificates (certificates <C21><C23><C24>, Figure 5) necessary to carry out this validation is called a *certification path.*

Two entities which act under the same policy always find a common node, which at worst will be the root of the PCAS node. When two entities act under different policies, they will only find a common node if a mapping exists between both policies.

The possibility of detecting a common node in the organization model allows the establishment of trust to be done automatically based on information contained in the certificates. When this common node does not exist, an entity can decide whether or not it trusts another entity by examining the policy under which the other entity acts.

One of the problems that all models of CA organization pose is that the finding the certificates that form the certification path between two entities is not trivial.

In the presented model the organization is based on the policies, therefore, an entity may have several certificates, one for each different policy. In order to locate the certification path between two entities, it is necessary to know the policy of both entities. This implies that the certification path that goes from each entity to its GCA must be known.

Each entity's certificate must contain information which assists in locating the certificates that form part of the certification path that joins the entity with the GCA. Version 3 of the certificate includes, with this aim, the fields *authorityKeyIdentifier* and *subjectKeyIdentifier* within the group *Key and Policy Information Extensions.* The version in [9] is as follows:

```
authorityKeyIdentifier EXTENSION ::= {
       SYNTAX         AuthorityKeyIdentifier
       IDENTIFIED BY  { id-ce 35} }
AuthorityKeyIdentifier ::= SEQUENCE {
 keyIdentifier [0] KeyIdentifier       OPTIONAL,
 authorityCertIssuer  [1] GeneralNames OPTIONAL,
 authorityCertSerialNumber
              [2]CertificateSerialNumber
                                       OPTIONAL}
WITH COMPONENTS{...,authorityCertIssuer PRESENT,
            authorityCertSerialNumber PRESENT}|
WITH COMPONENTS{...,authorityCertIssuer ABSENT,
            authorityCertSerialNumber ABSENT})
KeyIdentifier ::= OCTECT STRING
subjectKeyIdentifier EXTENSION ::= {
       SYNTAX         SubjectKeyIdentifier
       IDENTIFIED BY  { id-ce 14} }
SubjectKeyIdentifier ::= KeyIdentifier
```

With these two fields, clear identification of the public key and the issuing CA certificate, which are necessary for the verification of one's own certificate's signature, is achieved. Using these fields, all the certificates of the certification path which go from the entity to its GCA can be found progressively.

In this model it is not necessary to obtain all the certificates of the certification path of two entities to their respective GCA, in order to obtain the certification path from the certificate being validated to a common trust point. In the model it is only necessary to have information on which certificates are needed to establish the common trust point, but not the certificates themselves. Considering the fact that the certificate finding process is costly both in time and resources, the authors propose defining new certificate extensions that include references to the certificates that form the certification path from the certificate issuer to the corresponding GCA (Figure 5). The idea of including the identification of this certification path was first presented by the authors in October 1994 [3], and now they have adapted it to the latest format of the version 3 certificate, using the following fields:

```
certificationPathIdentifier EXTENSION ::= {
       SYNTAX CertificationPathIdentifier
       IDENTIFIED BY  { id-ce 12} }
CertificationPathIdentifier ::= SEQUENCE OF
                      AuthorityKeyIdentifier
AuthorityKeyIdentifier ::= SEQUENCE {
 autyType          [0] CertificateType,
 autyKeyIdentifier [1] KeyIdentifier OPTIONAL,
 autyCertIssuer    [2] GeneralNames,
 autyCertSerialNumber [3]
                       CertificateSerialNumber}
CertificateType ::= BIT STRING {
       groupAuthority        (0),
       subgroupAuthority     (1),
       certificationAuthority(2),
       user                  (3) }
KeyIdentifier ::= OCTECT STRING
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF
                  GeneralName
GeneralName ::= CHOICE {
 otherName     [0]     INSTANCE OF OTHER-NAME,
 rfc822Name    [1]     IA5String,
 dNSName       [2]     IA5String,
 x400Address   [3]     ORAddress,
 directoryName [4]     Name,
 ediPartyName  [5]     EDIPartyName,
 uniformResorurceIdentifier [6]IA5String,
 iPAddress     [7]     OCTECT STRING,
 registeredID  [8]     OBJECT IDENTIFIER }
OTHER-NAME ::= TYPE_IDENTIFIER
EDIPartyName ::= SEQUENCE {
 nameAssigner  [0]     DirectoryString OPTIONAL,
 partyName     [1]     DirectoryString }
newSubjectKeyIdentifier EXTENSION ::= {
       SYNTAX         NewSubjectKeyIdentifier
       IDENTIFIED BY  { id-ce 13} }
NewSubjectKeyIdentifier ::= SEQUENCE {
       subjectType     [0] CertificateType,
       subjectKeyIdent [1] KeyIdentifier }
id-ce-certificationPathIdentifier
              OBJECT IDENTIFIER ::= {id-ce 12}
id-ce-NewSubjectKeyIdentifier
              OBJECT IDENTIFIER ::= {id-ce 13}
```

The *authorityKeyIdentifier* extension field is replaced by the *certificationPathIdentifier* extension field and the *subjectKeyIdentifier* extension field is replaced by the *newSubject*Key*Identifier* extension field. In both new fields, the type of certificate, information which is necessary in order to automatically establish the trust point, is added. With this modification the *issuerAltName* extension field of the *Certificate Subject and Certificate Issuer Attributes Extensions* group is no longer necessary. For this reason this group now only includes *extensions* related to the subject.

4. Conclusions and future work

The wide acceptance of the X.509 certificate in different proposed applications of security services on telematic networks has meant that *The Directory - Authentication Framework* [7] is a necessary reference point in the realization of complex security environments. One of the greatest problems which arises is the establishment of a general organization model of Certification Authorities which can adapt to the needs of different groups which manage diverse types of information that needs to be securely transferred. The automation of the identification process and in the treatment of the different security policies is the next goal. The new version of the certificate, on whose definition Internet [5], ITU and ISO [8] [9] are working, already takes into account mechanisms to identify the policies using automatons. The main line of work which the authors' work group proposes for the future is the definition and formal specification of the policies that can be automatically managed in open environments.

References

[1] **American National Standards Institute.** *American National Standard X9.57. Public Key Cryptography for the Financial Services Industry: Certificate Management*. Draft Standard. 1996.

[2] **Berge, N.** *UNINETT PCA policy Statements*. Network WG RFC 1875 (Informational). Norwegian Computing Center, Dec. 1995.

[3] **Carracedo, J. and Gómez, A. (Eds.)** *COST225 Secure Communications.* Final Report. Brussels: COST Telecommunications Secretariat, Mar. 1995.

[4] **Housley, R., Ford, W., Solo, D.** *Internet Public Key Infrastructure. Part I: X.509 Certificate and CRL Profile*. Internet Draft. PKIX WG, Feb. 1996.

[5] **Internet.** *Privacy Enhanced Mail (PEM) for Internet Electronic Mail: RFC 1421, 1422, 1423 and 1424*. IAB Privacy Task Force, Feb. 1993.

[6] **ISO/IEC.** *Information Processing Systems. OSI Reference Model - Part 2: Security Architecture*. ISO/IEC International Standard 7498-2, Jul. 1988.

[7] **ISO/IEC and ITU.** *Information Technology - OSI. The Directory - Authentication Framework*. ISO/IEC International Standard 9594-8, ITU X.509, Dec. 1991.

[8] **ISO/IEC and ITU.** *Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 9594-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions.* ISO/IEC JTC 1/SC 21/WG4 and ITU-T Q15/7 Collaborative Editing Meeting on the Directory, Ottawa, Canada, Jul. 1995.

[9] **ISO/IEC and ITU.** *Final text of Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 9594-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions*. Final draft (June 30). ISO/IEC JTC 1/SC 21/WG 4 and ITU-T Q15/7 Collaborative Editing Meeting on the Directory, Geneva, Apr. 1996.

[10] **National Institute of Standard and Technology (NIST).** *Public Key Infrastructure Study*. Final Report, Gaithersburg, MD, Apr. 1994.

[11] **Roe, M. et al.** *PASSWORD R1.1: Service Requirements*. Aug. 1992.

[12] **Rivest, R., Shamir, A. and Adleman, L.** *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Comm. of the ACM, Vol. 21, No. 2 (pp. 120-126), Feb. 1978.

[13] **Visa and MasterCard.** *Secure Electronic Transaction. Book 2: Technical Specifications. Part I: System Design Considerations. Part II: Message Structure. Part III: Certificate Management. Part IV: Payment System. Part V: Appendices*. DRAFT for testing, San Francisco, Jun. 1996.

[14] **Zimmermann, P.** *Pretty Good Privacy (PGP) Public Key Encription for the Masses. PGP User's Guide Version 2.6.1*., Aug. 1994.